

Table of Contents

Colophon	1
Acknowledgments	11
Preface	13
History and theory behind Nim	14
Who is this for	15
Structure of the book	15
Part I: Introduction to Nim via graphics	17
1. Introduction	19
2. Drawing a line	21
2.1. Drawing horizontal and vertical lines	21
2.1.1. Drawing a line using one point, length, direction	22
2.1.2. Drawing a line by specifying start and end	24
3. Rendering Text	25
4. Sequences	27
5. Parameter passing and mutability	29
6. Let vs Var	31
7. Iterators	33
7.1. Yield	33
8. Generics	35
9. Templates	39
10. Macros	43
Part II: Nim language specification	45
11. Basic terms	47
12. Lexical analysis	49
12.1. Notation used in this chapter	49
12.2. Indentation	50

12.3. Comments	51
12.4. Multiline comments	52
12.5. Identifiers & Keywords	52
12.6. Identifier equality	53
12.7. String literals	54
12.8. Triple quoted string literals	55
12.9. Raw string literals	55
12.10. Generalized raw string literals	56
12.11. Character literals	56
12.12. Numeric Literals	58
12.12.1. Custom Numeric Literals	60
12.13. Operators	61
12.14. Other tokens	62
12.15. Unicode Operators	62
13. Syntax	63
13.1. Associativity	63
13.1.1. Precedence	63
13.2. Dot-like operators	65
14. Declarations and scope rules	67
15. Modules	69
15.1. Export marker	69
15.2. Module processing	69
15.3. Import statement	70
15.4. Include statement	71
15.5. Module names in imports	71
15.6. Collective imports from a directory	72
15.7. Pseudo import/include paths	72
15.8. From import statement	73
15.9. Export statement	73
15.10. Scope rules	74
15.10.1. Block scope	74
15.10.2. Tuple or object scope	74
15.10.3. Module scope	74
16. Type system	77
16.1. Ordinal types	77

16.2. Pre-defined integer types	78
16.3. Integer literals	79
16.4. Subrange types	80
16.5. Pre-defined floating-point types	80
16.5.1. Nan and Inf checks	81
16.6. Boolean type	82
16.7. Character type	83
16.8. Enumeration types	83
16.9. Overloadable enum field names	85
16.10. String type	86
16.11. cstring type	87
16.12. Structured types	88
16.13. Array and sequence types	88
16.14. Open arrays	90
16.15. Varargs	90
16.16. Unchecked arrays	91
16.17. Tuples and object types	92
16.18. <code>fields</code> and <code>fieldPairs</code> iterators	94
16.19. Object construction	95
16.20. Object variants	95
16.21. <code>cast uncheckedAssign</code>	98
16.22. Set type	98
16.22.1. Bit fields	100
16.23. Reference and pointer types	100
16.24. Nil	102
16.25. Procedural type	103
16.26. Calling conventions	103
16.27. Distinct type	105
16.27.1. <code>borrow</code> annotation	106
16.28. Auto type	106
16.29. <code>static[T]</code>	107
16.30. <code>typedesc[T]</code>	108
16.31. <code>typeof</code>	110
17. Type relations	113
17.1. Type equality	113

17.2. Subtype relation	113
17.3. Convertible relation	114
17.4. Assignment compatibility	116
18. Constant expressions	117
19. Overload resolution	119
19.1. Overloading based on 'var T'	122
19.2. Lazy type resolution for untyped	122
19.3. Varargs matching	123
19.4. Overload disambiguation	123
20. Statements and expressions	125
20.1. Statement list expression	125
20.2. Discard statement	125
20.3. Void context	126
20.4. Var statement	127
20.5. Let statement	129
20.6. Tuple unpacking	129
20.7. Const statement	130
20.8. Type section	130
20.9. Static statement/expression	132
20.10. If statement	133
20.11. Case statement	134
20.12. When statement	135
20.13. Return statement	136
20.14. Yield statement	137
20.15. Block statement	137
20.16. Break statement	137
20.17. While statement	138
20.18. Continue statement	138
20.19. Using statement	139
20.20. If expression	139
20.21. When expression	140
20.22. Case expression	140
20.23. Block expression	141
20.24. Table constructor	141
20.25. Type conversions	142

20.26. Type casts	142
20.27. The addr operator	143
20.28. The unsafeAddr operator	143
21. Procedures	145
21.1. Method call syntax	147
21.2. Properties	148
21.3. Indexing	149
21.4. Command invocation syntax	149
21.5. Closures	150
21.5.1. Creating closures in loops	150
21.6. Anonymous procs	151
21.7. Func	151
21.8. Non-overloadable built-ins	152
21.9. Var parameters	152
21.10. Var return type	153
22. Methods	155
22.1. Static method calls via procCall	156
23. Iterators and the for statement	157
23.1. Implicit items/pairs invocations	158
23.2. First-class iterators	159
24. User definable conversions	163
24.1. Converters	163
25. Exception handling	165
25.1. Try statement	165
25.2. Try expression	166
25.3. Except clauses	167
25.4. Defer statement	167
25.5. Exception hierarchy	169
25.6. Raise statement	169
26. Effect system	171
26.1. Exception tracking	171
26.2. EffectsOf annotation	173
26.3. Tag tracking	174
26.4. Side effects	175
26.5. GC safety effect	175

27. Generics	177
27.1. Is operator	178
27.2. Type Classes	178
27.3. Implicit generics	180
27.4. Generic inference restrictions	182
27.5. Symbol lookup in generics	183
27.5.1. Open and Closed symbols	183
27.6. Mixin statement	183
27.7. Bind statement	184
27.8. Delegating bind statements	184
27.9. Templates	185
27.10. Typed vs untyped parameters	186
27.11. Passing a code block to a template	186
27.12. Varargs of untyped	188
27.13. Symbol binding in templates	188
27.14. Identifier construction	189
27.15. Template parameter lookup rules	189
27.16. Hygiene in templates	190
27.16.1. Inject and gensym	191
27.17. Method call syntax limitations	192
28. Macros	193
28.1. Macros API	193
28.2. BindSym	195
28.3. For loop macros	196
29. Lifetime-tracking hooks	199
29.1. <code>=destroy</code> hook	199
29.2. <code>=wasMoved</code> hook	200
29.3. <code>=sink</code> hook	200
29.4. <code>=copy</code> hook	201
29.5. <code>=dup</code> hook	201
29.6. <code>=trace</code> hook	202
29.7. Move semantics	203
29.8. Swap	203
29.9. Sink parameters	204
29.10. Rewrite rules	205

29.11. Object and array construction	206
29.12. Destructor removal	206
29.13. Self assignments	206
29.14. Lent type	208
29.15. The <code>.cursor</code> annotation	208
29.16. Cursor inference / copy elision	209
29.17. Hook lifting	210
29.18. Hook generation	210
29.19. <code>nodestry</code> pragma	211
29.20. Copy on write	211
29.21. Practice	212
30. Strict funcs	217
31. View types	219
31.1. Path expressions	221
31.2. Start of a borrow	223
31.3. End of a borrow	223
31.4. Reborrows	223
Part III: Mastering Macros	225
32. Introduction	227
33. AST introspection	229
33.1. Typed vs untyped ASTs	230
34. AST creation	233
35. Collect macro	235
36. <code>strformat</code>	239
37. <code>strscans</code>	243
38. HTML trees	247
39. Advice	251
Part IV: Mastering Parallelism	253
40. Introduction	255
41. Threading	257
41.1. <code>createThread</code>	257
41.2. Single worker, single channel	258
41.3. Multiple workers, single channel	260
42. <code>spawn</code>	261
42.1. Return values	262

42.2. Sharing memory	264
43. Isolated data	267
44. Smart pointers	271
45. Parallel for each and reduce	273
45.1. <code>parMap</code>	274
45.2. <code>parReduce</code>	276
45.3. <code>parFind</code>	278
46. Final Advice	281
46.1. What to avoid	281
46.2. What to use	281
Appendix A: Grammar	283
Appendix B: Nim standard library cheat sheet	289
B.1. Integers	290
B.2. Strings	291
B.3. Sequences	292
B.4. Bit sets	293
B.5. Hashes	294
B.6. Hash sets	295
B.7. Hash tables	297
B.8. Optionals	299
B.9. String formatting	300
B.10. Algorithms	301
B.11. OS	303
B.12. JSON	305
B.13. Unicode	306
Index	307